13th International Workshop on OpenCL and SYCL

IWOCL 2025

Adapting the LLVM SPIR-V Backend for use in SYCL implementations

Alexey Sachkov, Intel

Vyacheslav Levytskyy, Intel.

April 7-11, 2025 | Heidelberg, Germany | iwocl.org

KHRONOS GROUP

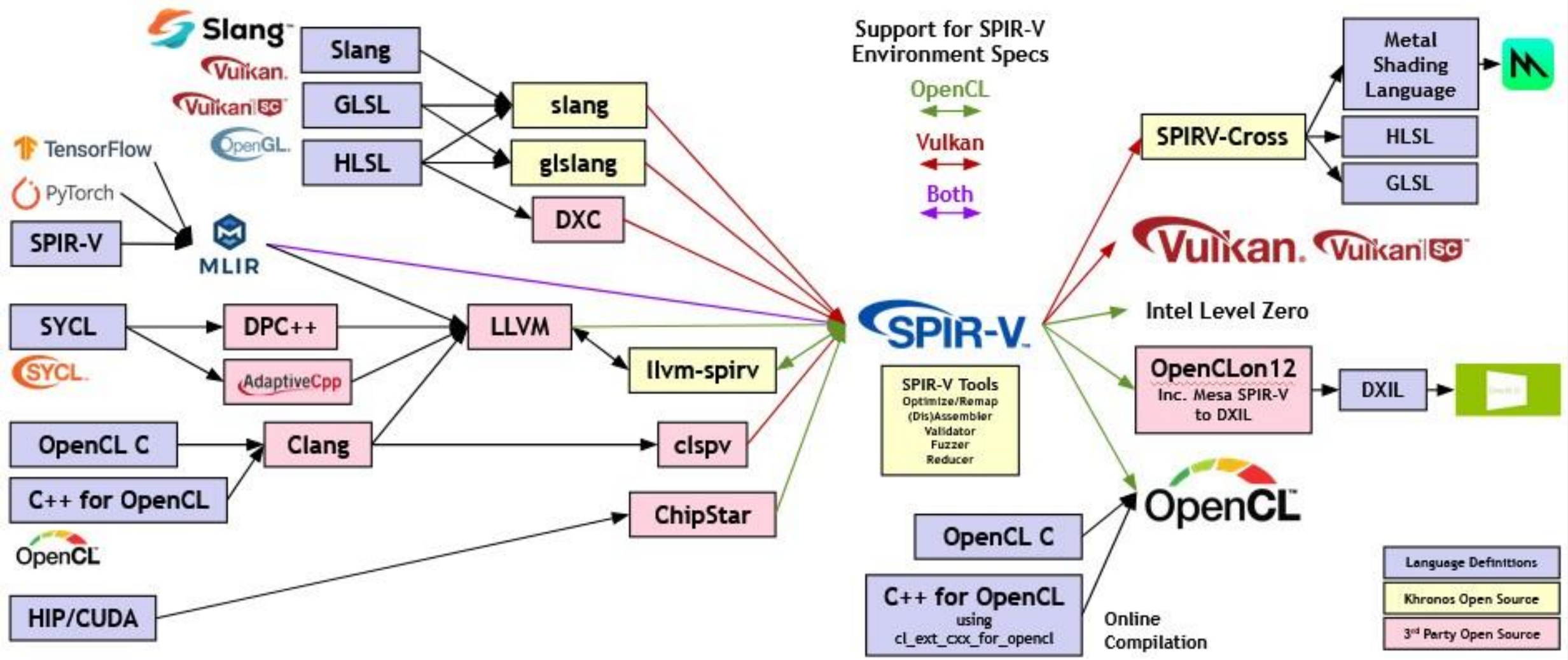
- Brief recap of key concepts
  - SPIR-V itself
- How and why, SPIR-V is used in SYCL (and other places)
  - Problems with the existing approach
- SPIR-V backend today
  - What is it and why does it exist
  - Comparison with the translator
  - State of things: progress and quality
  - Technical challenges
- Summary / Next steps

What is SPIR-V and Why It Matters

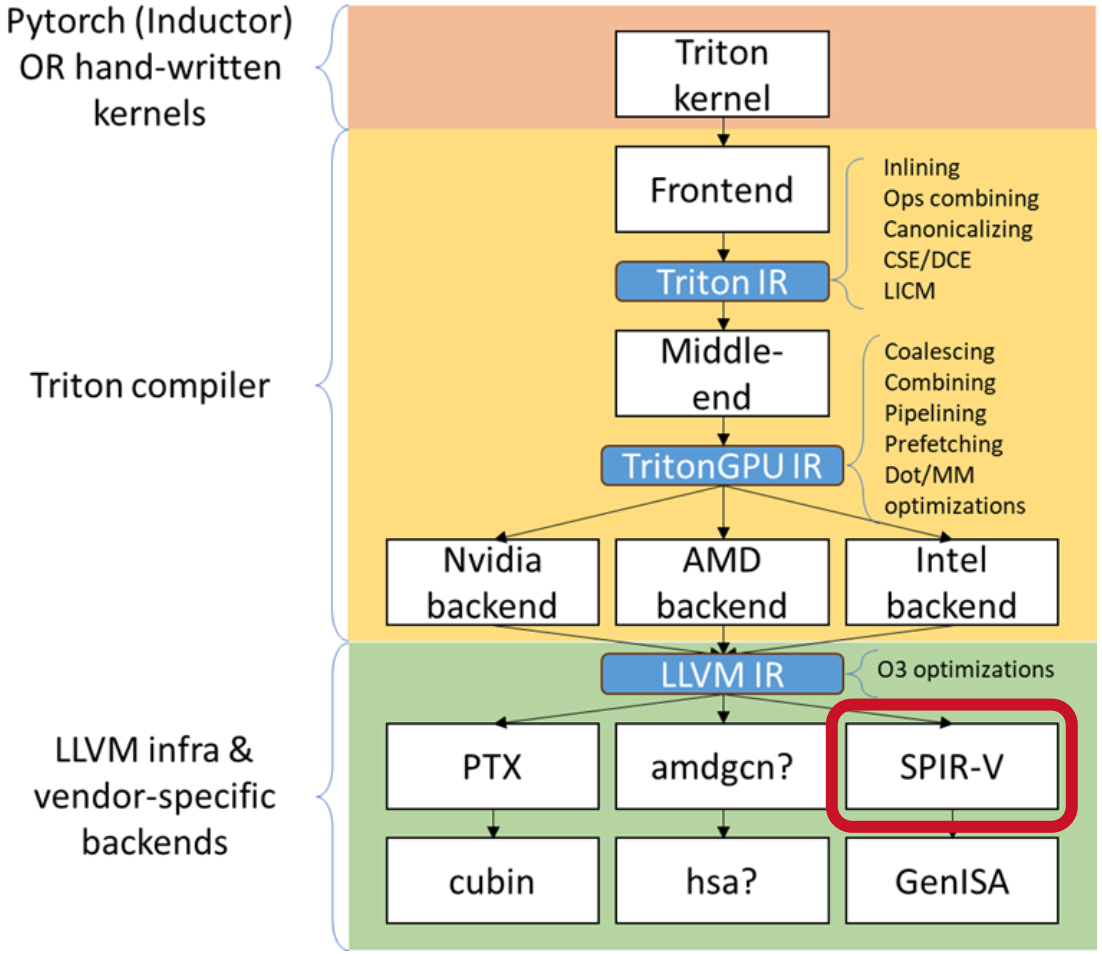
- SPIR-V is a binary intermediate language for representing graphical-shader stages and compute kernels
  - Both and IR and portable binary format serving as a programming interface for heterogeneous accelerators
  - Rich ecosystem of high-level languages and APIs
    - OpenCL, SYCL, GLSL, HLSL, Vulkan, OpenGL
  - Cross-vendor unifying intermediate representation

- The specification is defined by the Khronos Group
  - Vendors can add extension and client API environment specifications

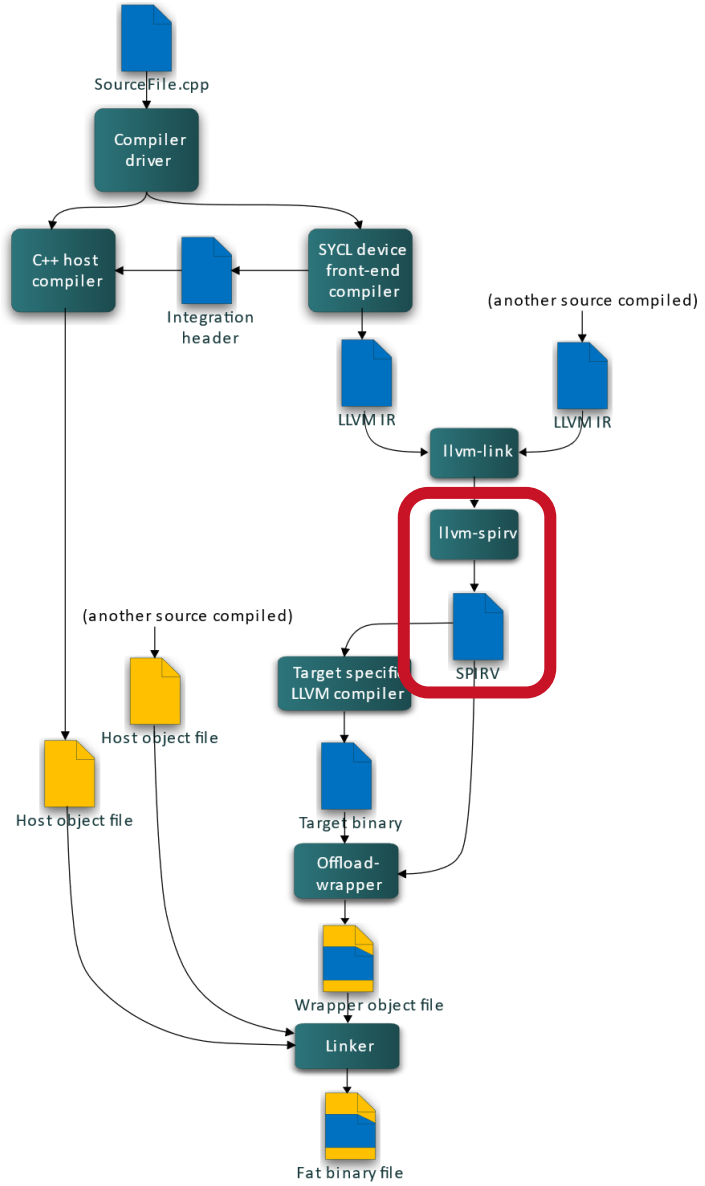
SPIR-V Language Ecosystem: khronos.org



SPIR-V and Heterogeneous Computing



intel/intel-xpu-backend-for-triton: [Architecture]



intel/llvm: [Compiler and Runtime architecture design]

SPIRV-LLVM-Translator

- Tool and a library for bi-directional translation between SPIR-V and LLVM IR
  - Focuses on support of compute kernels (OpenCL, Level Zero), i.e. `Kernel` capability
  - Doesn't support graphic/shader SPIR-V flavor, i.e. `Shader` capability

- Outside of the llvm/llvm-project tree!
  - Harder to integrate into a project
  - Extra effort to keep it in sync with the upstream LLVM
  - See also SPIR-V support in LLVM and Clang, 2021 LLVM Developer's Meeting

  - Multiple failed attempts
    - Integration through call to an external binary wasn't the ideal way: 0045d01a

SPIR-V Backend to the Rescue

- An official LLVM target that provides code generation for the SPIR-V format
  - It converts LLVM IR into SPIR-V either in text, or binary form
  - Uses LLVM code generation utilities and frameworks
  - Provides an alternative to the SPIRV-LLVM-Translator for LLVM IR -> SPIR-V path

- Why is it important?
  - The ultimate goal of getting first-class SPIR-V support into LLVM is achieved!
  - Being in LLVM makes everyone else in LLVM responsible for ***not*** breaking you
  - Works out-of-the box without external dependencies
  - Opens lots of opportunities for everyone:
    - Heterogeneous computing (e.g. OpenCL, SYCL)
    - AI (e.g. OpenAI Triton backend for Intel GPUs)
    - Other programming models, including graphical-shader stages (e.g. HLSL, Vulkan)

SPIR-V Backend to the Rescue

- It is being developed by engineers from companies-members of Khronos Group
  - Intel, Microsoft, Google
- It is built to serve both compute kernels and graphic shaders developers equally
- The team closely collaborates with OpenCL, SPIR and Vulkan Khronos working groups
  - To ensure that backend development aligns with the needs of the broader community

- Already has users outside of Khronos APIs:
  - DirectXShaderCompiler – the reference compiler for HLSL
    - Adds SPIR-V generation via the backend to enable HLSL as a frontend for Vulkan
  - Microsoft announced that DirectX will be adopting SPIR-V as an IR format of the future
  - In general, a good candidate for a cross-vendor unifying IR for smaller hardware vendors

SPIR-V Backend vs SPIRV-LLVM-Translator

- The backend is good enough to pass SYCL & OpenCL CTS tests with it
  - It allows to specify target SPIR-V version, as well as select extensions that can be used
- The main difference is in the list of supported extensions, built-ins and intrinsics
  - The backend is behind the translator here
- Missing features
  - Support for NonSemantic Shader DebugInfo instructions
  - SPV_INTEL_joint_matrix is implemented, but haven't been properly tested with SYCL
  - Some features available in Intel's SYCL implementation are intentionally unsupported
    - FPGA-specific things, for example

SPIR-V Backend Progress in 2024/2025: Serious Uplift

- Massive refactoring and new features
  - Support for 36 SPIR-V extensions was added, and more to come
  - Revisited and uplifted interaction with the LLVM
    - Rework of instruction patterns and virtual register types and classes
    - Improved interaction with GlobalISel and Machine Verifier
  - Ramped up support for different optimization levels
    - Everything works with -O2/-O3 now, not only -O0

- Identified some bugs in the SPIRV-LLVM-Translator
  - And provided fixes, of course!

- Established good contact with the LLVM community

SPIR-V Backend: Quality Assurance

- SPIR-V Backend test suite
  - Constantly growing
    - 600+ tests today
  - Fragments of bigger test suites are being embedded to ensure validity of the output

- Functional testing
  - A comprehensive set of use cases along several axis:
    - SPIR-V features, type inference and aggregates, compute-centric and graphical use cases
  - SPIR-V validator is being used as well!
  - Everything is automated using GitHub Actions

SPIR-V Backend: Quality Assurance

- External testing
  - SYCL-CTS are being regularly run through SPIR-V backend in intel/llvm GitHub repository
  - Intel XPU backend for Triton project has CI workflows using SPIR-V backend
  - It is sometimes hard to keep the pass rate stable:
    - Lots of moving pieces, i.e. all components are changing rapidly
    - Some time lag is involved between different projects (including upstream/downstream)

- Non-functional testing
  - Compile-time performance tracking
    - Two major reworks brought 25x speedup between Dec 2024 and Mar 2025
  - Run-time performance of the produced SPIR-V
    - Doing some *_preliminary_* small scope testing (GROMACS on PVC GPU) performance is on par with the SPIRV-LLVM-Translator

SPIR-V Backend: More Integrations Work in Progress

- Intel Extensions for OpenXLA
  - Run JAX models on Intel GPUs
- Intel MLIR-based Graph Compiler
- Intel's SYCL compiler end-to-end test suite
  - Stable high pass rates around 94-99%
  - Not everything is supported yet (intentionally, or temporarily)

SPIR-V Backend: Community

- Maintained by a team of dedicated contributors
- Regular LLVM SPIR-V working group meetings
  - See LLVM: Getting Involved
- Regular conference coverage
  - LLVM Developer's Conference 2022 (video, slides) and 2024 (video, slides)
  - IWOCL 2025 (this talk)
  - LLVM Developer's Conference 2025 (next week)
- Comprehensive documentation
  - User Guide for SPIR-V Target, SPIR-V support in Clang

SPIR-V Backend Technical Challenges: Causes

- Concepts of the SPIR-V language
  - Cannot be easily represented in LLVM's Machine IR
  - Not fully supported by the standard GlobalISel translation schema
  - Have definitions conflicting with Machine Verifier requirements

- SPIR-V is a semantically rich language
  - About the same level as LLVM IR
  - But LLVM CodeGen expects its target to be a low-level language

SPIR-V Backend Technical Challenges: Examples

- Semantics mismatch
  - There is no way to encode `if (Cond) then Stmt` logic, only full `if-then-else` is supported
    - By OpBranchConditional in SPIR-V
  - `OpLabel` is not a traditional assembler label as in LLVM's ASMPrinter

- Aggregates lowering
  - Mismatch between LLVM's virtual register and SPIR-V's identifier concepts
  - LLVM low level types are limited by scalars, pointer and vector types
    - Aggregates are decomposed, but SPIR-V supports aggregates!
  - The backend emits internal intrinsics to map virtual register to original aggregate values
    - To preserve original names and data types

- LLVM uses opaque pointers
  - But SPIR-V's pointers are typed
    - And consumers are quite sensitive to those pointer type changes

SPIR-V Backend: Summary

- Is an official (***non***-experimental) target since January 2025 (RFC)
  - Functionally- & Performance-wise isn't drastically different from the SPIRV-LLVM-Translator

- Grows faster than the SPIRV-LLVM-Translator
  - Common core benefits from contributions of compute and shader flavors of SPIR-V
  - Lots of different use cases and contributors

- Presence in the upstream ensures tighter integration, making it easier to influence LLVM
  - That also means that OpenCL & SYCL have better influence through the backend
  - An opportunity to harden SPIR-V as an industry standard interchange format
    - For both compute and graphics

Try it yourself!

Next steps

- Continue to enhance the SPIR-V backend
  - Support for more extensions, built-ins, intrinsics
  - Be aligned with the SPIRV-LLVM-Translator
- Incorporate feedback from SPIR-V backend users
  - And grow userbase
- Further improve quality and amount of testing
  - Automate non-functional testing
  - Port over reaming relevant parts of the SPIRV-LLVM-Translator testing

- Release oneAPI DPC++ compiler with SPIR-V Backend enabled by default
  - Aim to flip the switch this year
    - Still allowing to fallback to the SPIRV-LLVM-Translator if necessary

Thank you!

Questions?


intel